Ensemble of Classifiers Approach for NDT Data Fusion

Devi Parikh, Min T. Kim, Joseph Oagaro, Shreekanth Mandayam, and Robi Polikar*
Department of Electrical and Computer Engineering,
Rowan University, Glassboro, NJ 08028, USA
{parikh55, kimm18, oagaro00}@students.rowan.edu, {shreek, polikar}@rowan.edu

Abstract: *Several measurement modalities have been developed over the years for various nondestructive testing and evaluation (NDT&E) applications, such as ultrasonic, magnetic flux leakage, and eddy current testing, all of which have been used extensively in pipeline defect identification. While it is generally believed that different testing modalities provide complementary information, only a single testing modality is typically used for a given application. This is in part due to lack of effective, computationally feasible data fusion algorithms that are applicable to NDT&E signals. Such an algorithm capable of data fusion can combine information from two or more different sources of data, giving more insight and confidence to the data analysis than a decision that would otherwise be based on either of the sources alone. Learn++, previously introduced as an incremental learning algorithm, was applied to a NDT&E data fusion application. Specifically, we generated two ensembles of classifiers, one trained on ultrasonic signals, and the other on corresponding magnetic flux leakage signals obtained from stainless steal samples that contained five classes of discontinuities: crack, pitting, weld, mechanical damage, and no discontinuity. We have observed that the prediction ability of the automated classification system, as measured by the accuracy and reliability of the classification performance on validation data, was significantly improved when the two data sources were combined using Learn++.*

Keywords: Data fusion, combining classifiers, ensemble systems, incremental learning, Learn++

1 Introduction:

We have previously introduced Learn++ as an effective automated classification algorithm that is capable of learning incrementally from new data that may later become available, after a classification system has already been created with the initially available data. The algorithm is based on generating an ensemble of classifiers and appropriately combining the outputs of these classifiers. We have recently discovered that the ensemble of classifiers approach used for incremental learning is directly applicable for data fusion applications. This is because data fusion also involves learning from additional data, albeit with a different set of features. Our approach – which can be considered as a decision level fusion – is then to employ an ensemble of classifiers strategically generated by using all of the data sources available such that the complementary pieces of information provided by different datasets are best utilized. The classifier outputs predicting a specific defect type are then combined through weighted majority voting of the classifier outputs, where the weights are determined based on the training / previous performance of each classifier. The defect class that receives the highest weighted vote is finally selected by the ensemble system.

1.1 Incremental Learning and Data Fusion

Classification algorithms usually require availability of an adequate and representative set of training data to generate an appropriate decision boundary and provide a satisfactory generalization performance. This is particularly true if an ensemble approach of classification is used and the classifiers are combined using trainable rules such as weighted majority voting, weighted sum rule and weighted product rules, as opposed to fixed rules such as the simple sum, product and majority voting rules [1]. However, acquisition of such data is expensive and time consuming, and consequently it is not uncommon for the entire data to become available gradually in small batches over a period of time. Furthermore, the datasets acquired in subsequent batches may introduce instances of new classes that were not present in previous datasets. In such settings, it is necessary for an existing classifier to be able to acquire the newly introduced knowledge without forgetting the previously learned information. The ability of a classifier to learn in this fashion is usually referred to as *incremental learning*.

It is well known that data available from multiple sources underlying the same phenomenon may contain complementary information. For instance, in non-destructive evaluation of pipelines, defect information may be obtained from eddy current, magnetic flux leakage images, ultrasonic scans, thermal imaging, etc. Intuitively, if such information from multiple sources can be appropriately combined, the performance of a classification system can be improved. A classification system, capable of combining information from multiple sources or from multiple feature sets, is said to be capable of performing data fusion. Consequently, both incremental learning and data fusion involve learning from different sets of data. In incremental learning the datasets may introduce new classes, whereas in data fusion the datasets may contain different features, indicating a conceptual similarity between incremental learning and data fusion.

1.2 Ensemble Approach for Incremental Learning

A multiple classifier system (MCS) combines an ensemble of generally weak and/or diverse classifiers. The diversity in the classifiers allows different decision boundaries to be generated by using slightly different training parameters, such as different training datasets. The intuition is that each classifier will make a different error, and strategically combining these classifiers can reduce total error. Thus, MCS takes advantage of the so-called *instability* of the weak classifier and in turn generates a strong classifier

[2-3]. Ensemble or MCS have attracted a great deal of attention over the last decade due to their reported superiority over single classifier systems on a variety of applications [3-5].

The ensemble approach has also been widely used with a variety of algorithms to improve the generalization performance of a classification system. However, using this approach to solve the problem of incremental learning has been mostly unexplored. In exploration of such an approach to the incremental learning problem, we have recently developed Learn++, and have shown that it is indeed capable of incrementally learning [5]. Recognizing the above mentioned conceptual similarity between incremental learning and data fusion, we now evaluate Learn++ on a real world data fusion problem. The approach and our preliminary results are presented in this paper.

1.3 Ensemble Approaches for Data Fusion

Several approaches have been developed for data fusion, for which ensemble approaches constitute a relatively new breed of algorithms. Traditional methods are generally based on probability theory, such as the Dempster-Schafer (DS) theory and its many variations. However, DS based algorithms require specific knowledge of the underlying probability distribution, which may not be readily available.

The majority of these algorithms have been developed in response to the needs of military applications, most notably target detection and tracking [6-8]. Ensemble approaches seek to provide a fresh and a more general solution for a broader spectrum of applications. Such approaches include simpler combination schemes such as majority vote, threshold voting, averaged Bayes classifier, maximum/minimum rules, and linear combinations of posterior probabilities [9,10]. More complex data fusion schemes are also widely used, including ensemble based variations of DS, neural and fuzzy systems, and stacked generalization [11 - 16].

2 Learn++

The novelty of Learn++ is its incremental learning capability. It can learn new information as and when new data become available, without forgetting the previously acquired knowledge and without requiring access to the previous data, hence without suffering from catastrophic forgetting [17]. Specifically, Learn++ generates an ensemble of relatively weak classifiers for each new database that becomes available, where the outputs of each individual classifier of the ensemble are combined through weighted majority voting to obtain the final classification.

Weak / diverse classifiers are trained on a subset of the training data, randomly selected from a dynamically updated distribution over the training data instances. This distribution is biased towards those instances that have not been properly learned or seen by the previous ensemble(s).

For each database, FS$_k$, $k=1,...,K$, comprised of a different *F*eature *S*et (obtained from the same particular application) that is submitted to Learn++, the inputs to the algorithm are (i) a sequence S_k of m_k training data instances $\boldsymbol{x_i}$ along with their correct labels y_i ; (ii) a supervised classification algorithm BaseClassifier, generating individual classifiers (henceforth, hypotheses); and (iii) an integer T_k, the number of classifiers to be generated for the k^{th} database.

The only requirement on the BaseClassifier algorithm is that it can obtain at least 50% correct classification performance on its own training dataset, so that a minimum reasonable performance can be expected from each classifier. Note that for a two-class problem, 50% performance is equivalent to random guessing. BaseClassifier can be any supervised classifier such as a multilayer perceptron, radial basis function, or a support vector machine. Their weakness can be controlled by adjusting their size and error goal with respect to the complexity of the problem. Sufficiently different decision boundaries can then be generated by these weak classifiers by training them with slightly different training datasets. It should be noted that most of the resources in generating a strong classifier are typically spent in fine-tuning the decision boundary. Since Learn++ requires only a rough estimate of the decision boundary from its weak classifiers, the expensive step of fine-tuning is avoided. This saves on computational time during training, and also helps prevent overfitting of the training data.

Each hypothesis h_t is trained on a different subset of the training data. This is achieved by initializing a set of weights for the training data, $\boldsymbol{w_t}$, and a distribution $\boldsymbol{D_t}$ obtained from $\boldsymbol{w_t}$. According to this distribution a training subset TR_t is drawn from the training data at the t^{th} iteration of the algorithm. The distribution $\boldsymbol{D_t}$ determines which instances of the training data are more likely to be selected into the training subset TR_t. Unless a priori information indicates otherwise, this distribution is initially set to be uniform, giving equal probability to each instance to be selected into the first training subset. At each subsequent iteration *t*, the weights previously adjusted at iteration *t-1* are normalized to ensure a legitimate distribution $\boldsymbol{D_t}$ (step 1).

Training subset TR_t is drawn according to $\boldsymbol{D_t}$ (step 2), and the weak classifier is trained on TR_t in step 3. A hypothesis h_t is generated by the t^{th} classifier, whose error ε_t, is computed on the entire (current) database S_k as the sum of the distribution weights of the misclassified instances (step 4)

$$\varepsilon_t = \sum_{i:h_t(x_i)\neq y_i} D_t(i) \tag{1}$$

As mentioned above, the error, as defined in Equation (1), is required to be no greater than 0.5 to ensure that a minimum reasonable performance can be expected from h_t. If this is the case, the hypothesis h_t is accepted and the error is normalized to obtain the normalized error.

$$\beta_t = \frac{\varepsilon_t}{1-\varepsilon_t}, 0 < \beta_t < 1 \tag{2}$$

If $\varepsilon_t \geq 0.5$ then the current hypothesis is discarded, and a new training subset is selected by returning to step 2. All *t* hypotheses generated thus far are then combined using a voting scheme to obtain a composite hypothesis H_t (step 5).

$$H_t = \arg\max_{y\in Y} \sum_{t:h_t(x)=y} \log\left(\frac{1}{\beta_t}\right) \tag{3}$$

The voting scheme used by Learn++ is not quite democratic. Each hypothesis is assigned a weight as the logarithm of the reciprocal of its normalized error. Therefore, those hypotheses with smaller training error, indicating better performances, are awarded with a higher voting weight and thus have more say in the final classification decision. The error of the composite hypothesis H_t is then computed in a similar fashion as the sum of the distribution weights of the instances that are misclassified by H_t (step 6)

$$E_t = \sum_{i:H_t(x_i)\neq y_i} D_t(i) = \sum_{i=1}^{m} D_t(i)[\![H_t(x_i) \neq y_i]\!] \tag{4}$$

where, | · | evaluates to 1, if the predicate holds true and 0 otherwise. The normalized composite error B_t is obtained as

$$B_t = \frac{E_t}{1-E_t}, 0 < B_t < 1 \tag{5}$$

which is then used for updating the distribution weights assigned to individual instances

$$w_{t+1}(i) = w_t(i) \times \begin{cases} \mathrm{B}_t, H_t(x_i) = y_i \\ \\ 1, \mathit{otherwise} \end{cases} \tag{6}$$

Equation (6) indicates that the distribution weights of the instances correctly classified by the composite hypothesis H_t are reduced by a factor of B_t ($0<B_t<1$). Effectively, this increases the weights of the misclassified instances making them more likely to be selected to the training subset of the next iteration. We note that this weight update rule, based on the performance of the current ensemble, facilitates incremental learning. This is because, when a new dataset is introduced (particularly with new classes), the existing ensemble (H_t) is bound to misclassify the instances that have not yet been properly learned, and hence the weights of these instances are increased, forcing the algorithm to focus on learning novel information introduced by the new data.

At any point, a final hypothesis H_{final} can be obtained by combining all hypotheses that have been generated thus far.

$$H_{final}(\mathbf{x}) = \arg\max_{y\in Y} \sum_{k=1}^{K} \sum_{t:h_t(\mathbf{x})=\mathbf{y}} \log\left(\frac{1}{\beta_{tk}}\right) \tag{7}$$

Figure 1 illustrates using Learn++ is a data fusion setting.

3 Results

While Learn++ was originally developed as an incremental learning algorithm, its ensemble structure allows it to be used in data fusion applications as well. This is because the algorithm can accept a new dataset even if it contains completely different features as compared to the data the algorithm has previously seen. When used in data fusion mode, Learn++ seeks to incrementally learn novel information content from databases that come from the same application but are composed of different features.

Implementing data fusion using Learn++ with the ensemble approach was tested on a real world application – identifying defects in pipelines using non-destructive techniques. Two datasets of different features were fused. The first was a set of Magnetic Flux Leakage (MFL) images, and the second was a set of Ultrasonic Testing (UT) images.

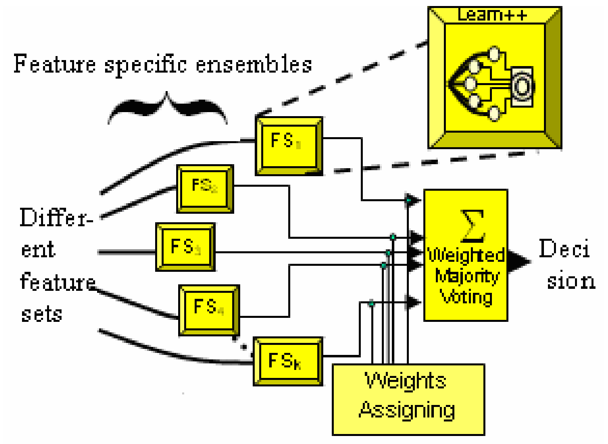

Figure 1. Block diagram of the ensemble based data fusion algorithm

Both modalities can be used to detect and identify defects in pipes. Illustrations of these images along with the type of defect they represent are shown in Figure 3.

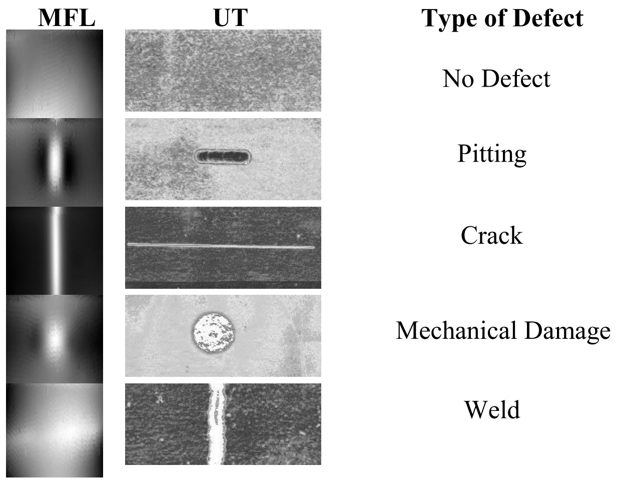

Figure 2. Sample MFL and UT images of defect types

The database consisted of 21 images from to a total of 5 classes: (i) No defect: 4 images; (ii) Pitting: 9 images; (iii) Crack: 4 images; (iv) Mechanical Damage: 4 images; (v) Weld: 4 images. Ten images (2 from each class) were used as the training data and the remaining 11 as the testing data. This distribution was kept constant for all dataset shuffles for different runs and to perform cross validation.

The Learn++ algorithm was run several times in data fusion mode with different combinations of the parameters such as the error goal and number of hidden layer nodes in the MLP networks. Classifiers were added to the ensemble until classification performance leveled off beyond a certain number of classifiers. The algorithm was run using a single hidden layer MLP as the base classifier with the following parameters (observed to be the optimum range based on prior experience): error goal: 0.05 ~ 0.08 in steps of 0.01, and number of hidden layer nodes: 5 ~ 45 in steps of 5. Every possible combination of the above parameters was used. Also, the experiment was duplicated using a different partition of the data (a different selection of the instances used for training and testing). Therefore, there were 36 generalization performance values for each partition of data, yielding a total of 72 generalization performances. The number of classifiers in an ensemble trained on each of the two feature sets could vary from 1 to 50 classifiers. The results obtained are summarized in Table 1.

Table 1: Comparing the data fusion performance to the individual performance of MFL (MFLp) and UT (UTp)

Data fusion performance combining two feature sets is	Proportions %
Greater than max (MFLP, UTP)	31.94
Equal to max (MFLP, UTP)	40.28
Equal to both MFLP and UTP	9.72
Between min (MFLP,UTP) max (MFLP,UTP)	6.94
Equal to min (MFLP, UTP)	8.33
Less than min (MFLP, UTP)	2.78
Total	100

The second column of Table 1 indicates the percentages of different comparisons between the data fusion performance and the individual MFL and UT performances, out of the 72 experiments. For example, in 31.94% (23 out of the 72) of the simulations, the data fusion performance was better than either of the individual MFL or UT performance. Adding the numbers in rows 5 and 6, it can be seen that the proportions of undesirable cases (when the data fusion performance is the lower of MFL and UT or worse than both) is about 11.11% of the times data fusion was performed (8 out of 72). The most desirable cases (when the data fusion performance is the higher of MFL and UT or better than both) are about 72.22% of the times (52 out of 72) data fusion was performed.

The above results are promising, but not optimum. The results were analyzed further and a few drawbacks in the algorithm were identified, such as the discrepancies between the number of classifiers from each feature set, as well as β assuming a zero value, and fixed. As a result of these modifications, the undesirable instances were eliminated and the desirable instances were observed about 97.2% of the times data fusion was performed.

4 Conclusions

Recognizing the conceptual similarities between incremental learning and data fusion, the Learn++ algorithm – originally developed for incremental learning – has been evaluated in a data fusion setting. The algorithm incrementally and sequentially learns data comprised of different sets of features by generating an ensemble of classifiers for each dataset, and then combining them through a weighted majority voting scheme. We have evaluated the algorithm on a real world data fusion application for pipeline defect identification, where the two different datasets consisted of UT and MFL signals of the same pipeline specimens. The results indicate that the Learn++ algorithm, when used to combine information contained in two datasets, performed significantly better then each of the testing modalities individually. Various statistical tests have been performed to establish this fact (not included here for space considerations). The ability of the algorithm to learn incrementally as well as to fuse different datasets to extract additional information not available in either dataset makes Learn++ a versatile algorithm. Further testing of the algorithm on additional real world and benchmark data is currently underway.

Acknowledgements

This material is based upon work supported by the National Science Foundation under Grant No. ECS-0239090.

References

[1] S. Prabhakar, A. K. Jain, "Decision level fusion in fingerprint verification," *Pattern Recognition*, vol. 35, pp. 861-874, 2001.
[2] L.K. Hansen and P. Salamon, "Neural network ensembles," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 12, no. 10, pp. 993-1001, 1990.
[3] T.G. Dietterich, "An experimental comparison of three methods for constructing ensembles of decision trees: Bagging, boosting and randomization," *Machine Learning*, vol. 40, no. 2, pp. 1-19, 2000
[4] T. Windeatt and F. Roli (eds), *Proc. 3rd Int. Workshop on Multiple Classifier Systems (MCS 2002),* LNCS vol. 2364, p. 1-15, Springer: New York, NY, 2002
[5] R. Polikar, L. Udpa, S. Udpa, V. Honavar, "Learn++: an incremental learning algorithm for supervised neural networks," *IEEE Trans Systems, Man and Cybernetics*, vol.31, no.4, pp.497-508, 2001.
[6] D. Hall and J. Llinas, "An introduction to multisensor data fusion," *IEEE Proceedings*, vol. 85, no. 1, 1997.
[7] D. Hall and J. Llinas (editors), *Handbook of multisensor data fusion*, CRC Press: Boca Raton, FL, 2001.
[8] L. A. Klein, Sensor *and Data Fusion Concepts and Applications*, SPIE Press, vol. TT35: Belingham, WA, 1999.
[9] J. Grim, J. Kittler, P. Pudil, and P. Somol, "Information analysis of multiple classifier fusion," *Proc. 2nd Intl Workshop on Multiple Classifier Systems*, LNCS vol. 2096, pp. 168-177, Springer: New York, NY, 2001.
[10] J. Kittler, M. Hatef, R.P. Duin, J. Matas, "On combining classifiers," *IEEE Trans on Pattern Analysis and Machine Intelligence,* vol. 20, no.3, pp. 226-239, 1998.
[11] L.O. Jimenez, A.M. Morales, A. Creus, "Classification of hyperdimensional data based on feature and decision fusion approaches using projection pursuit, majority voting and neural networks," *IEEE Trans Geoscience and Remote Sensors*, vol. 37, no. 3, pp 1360-1366, 1999.
[12] G.J. Briem, J.A. Benediktsson, and J.R. Sveinsson, "Use of multiple classifiers in classification of data from multiple data sources," *Proc. of IEEE Geoscience and Remote Sensor Symposium*, vol. 2, pp. 882-884, Sydney, Australia, 2001.
[13] F.M. Alkoot, J. Kittler. "Multiple expert system design by combined feature selection and probability level fusion," *Proc of the 3rd Intl Conf on FUSION 2000,* vol. 2, pp. 9-16, 2000
[14] D. Wolpert, "Stacked generalization," *Neural Networks*, vol. 2, pp 241-259, 1992.
[15] L.I. Kuncheva, "Switching between selection and fusion in combining classifiers: an experiment," *IEEE Trans. on Sys., Man and Cyber.*, vol. 32(B), no. 2, pp. 146-156, 2002.
[16] L.I. Kuncheva, "A theoretical study on six classifier fusion strategies, " *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 24, no. 2, pp. 281-286, 2002.
[17] R. French, "Catastrophic forgetting in connectionist networks," *Trends in Cognitive Sciences*, vol. 3, no.4, 1999.